JAIN SIP Tutorial
Serving the Developer Community

Phelim O'Doherty

Sun Microsystems

Mudumbai Ranganathan

NIST

JAIN SIP is the standardized Java interface to the Session Initiation Protocol for desktop and server applications.

JAIN SIP enables transaction stateless, transaction stateful and dialog stateful control over the protocol.

Presentation Outline

- What is SIP?

- Why create JAIN SIP?

- Introduction to JAIN SIP

- Developer Code Snippets

- Implementation Used-Cases

Session Initiation Protocol

- Session Initiation Protocol (SIP) is a signaling protocol for creating, modifying and destroying dialogs between multiple endpoints:
  - Request/response protocol (like HTTP, but peer-peer)
  - Simple and extensible
  - Designed for mobility (proxy/redirect servers)
  - Bi-directional authentication
  - Capability negotiation

- SIP is used for controlling the signaling that enables manipulates of sessions such as:
  - Instant Messaging sessions
  - Phone calls over the Internet
  - Gaming servers
  - Resource Location

SIP Functionality

- SIP supports five facets of establishing and terminating multimedia communications these include:
    - User location: determination of the end system to be used for communication.
    - User capabilities: determination of the media and media parameters to be used.
    - User availability: determination of the willingness of the called party to engage in communications.
    - Call setup: "ringing", establishment of call parameters at both called and calling party.
    - Call handling: including transfer and termination of calls.

Presentation Outline

- What is SIP?

- Why create JAIN SIP?

- Introduction to JAIN SIP

- Developer Code Snippets

- Implementation Used-Cases

Why Create JAIN SIP?

- SIP is an IETF specification that has been adopted by the communications industry in the form of 3GPP, 3GPP2, OMA and ITU.

- The IETF specification defines the SIP protocol in text format

- The SIP Community holds various interoperability events to ensure the credibility of the protocol.

- As a developer you are free to implement the protocol in any language, hence define your own interface for accessing the defined behavior of the protocol as outlined by the IETF standard.

- While IETF specification ensures interoperability between stacks, it doesn't address interoperability of applications across stacks.

- JAIN SIP satisfies this need in the Java programming language. It ensures *true* interoperability in that by utilizing the JAIN SIP specification you have interoperability between stacks and the interoperability of applications across stacks, often referred to as *application portability*.
    - Both stack interoperability and application portability are required in this new age of communication standards.

Presentation Outline

- What is SIP?

- Why create JAIN SIP?

- Introduction to JAIN SIP

- Developer Code Snippets

- Implementation Used-Cases

JAIN SIP

- The Java-standard interface to a SIP signaling stack.
  - Standardizes the interface to the stack.
  - Standardizes message interface.
  - Standardizes events and event semantics.
  - Application portability - verified via the TCK.

- Designed for developers who require powerful access to the SIP protocol.

- JAIN SIP can be utilized in a user agent, proxy, registrar or imbedded into a service container.

JAIN SIP Functionality

- JAIN SIP supports the SIP protocol functionality described in RFC 3261.

- JAIN SIP the following SIP extensions;
  - RFC 2976 allows for the carrying of session related control information that is generated during a session.
  - RFC 3262 provide information on progress of the request processing.
  - RFC 3265 the ability to request asynchronous notification of events.
  - RFC 3311 allows the caller or callee to provide updated session information before a final response.
  - RFC 3326 the ability to know why a SIP request was issued.
  - RFC 3428 allows the transfer of Instant Messages.
  - RFC 3515 requests that the recipient refer to a resource provided in the request.

JAIN SIP Object Architecture
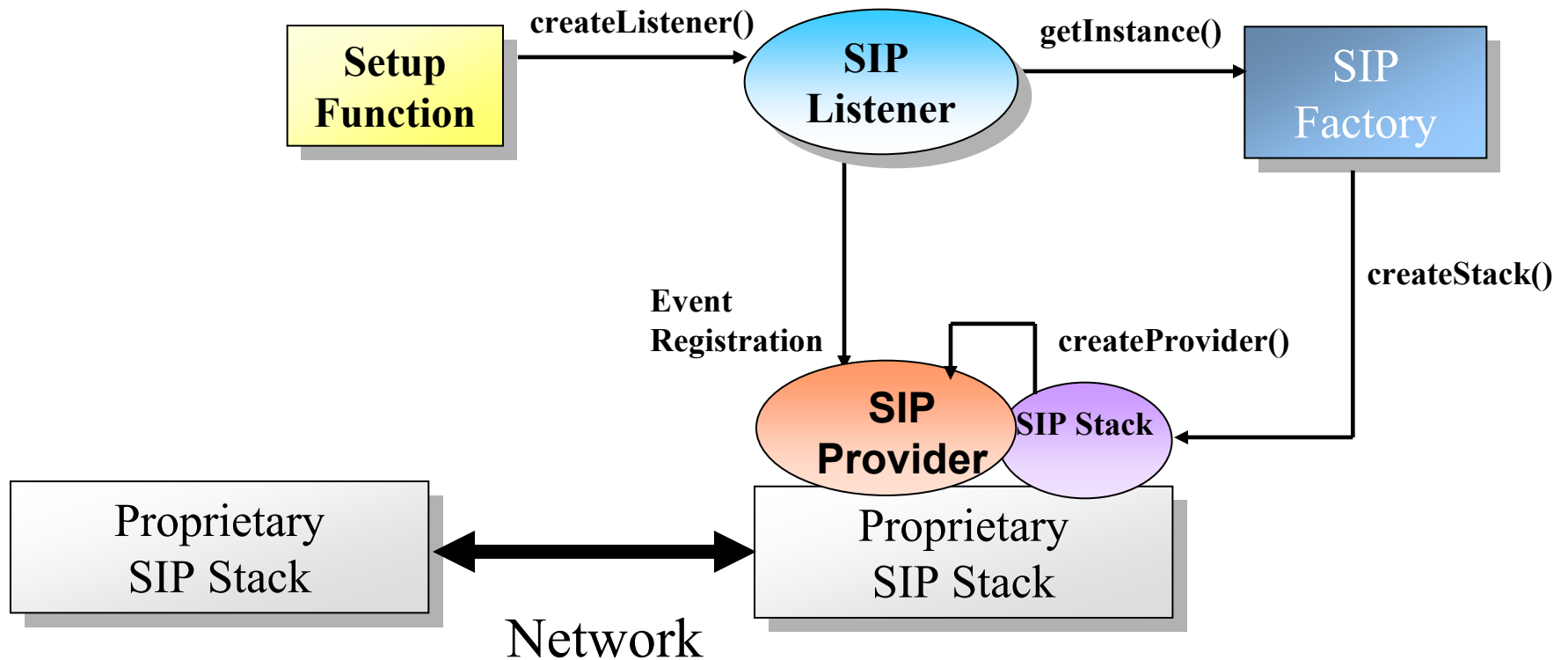
SipStack Interface

- Manages Listening Points and Providers.
- SipStack associated with an IP address.
  - Can have multiple Listening points.
- Application can have multiple SipStacks.
- Cannot be deleted once created.
- Instantiated by the SipFactory and initialized with a property set.
- `'javax.sip.*'` properties are reserved and names defined for stack configuration properties.
- Defines retransmission settings.
- Defines router information.

Retransmissions

- JAIN SIP provides a convenience function that ensures all retransmissions are handled by the JAIN SIP implementation.
    - Reduces complexity for applications acting as user agents.
    - Reduces complexity for integrating JAIN SIP as a base implementation for a SIP Servlet container or a JAIN SLEE implementation.

- Configured via Java properties on the SipStack Interface.
    - Default is off.

- The default handling of message retransmissions in JAIN SIP is dependent on the application.
    - Stateful proxy applications need not be concerned with retransmissions as these are handled by JAIN SIP.
    - Typically User Agent applications must handle retransmissions of ACK's and 2xx Responses.

Stack Properties

- IP_ADDRESS
  - Sets the IP Address of the SipStack. This property is mandatory.

- STACK_NAME
  - Sets a user friendly name to identify the underlying stack implementation. This property is mandatory.

- OUTBOUND_PROXY
  - Sets the outbound proxy of the SIP Stack.

- ROUTER_PATH
  - Sets the fully qualified classpath to the application supplied Router object that determines how to route messages before a dialog is established.

- EXTENSION_METHODS
  - This configuration value informs the underlying implementation of supported extension methods that create new dialog's.

- RETRANSMISSION_FILTER
  - A helper function for User Agents that enables the stack to handle retransmission of ACK Requests, 1XX and 2XX Responses to INVITE transactions for the application.

SipProvider Interface

- Register a SipListener to the SipProvider.
  - Notifies registered Listener of Events

- De-register a SipListener from the SipProvider.
  - Once de-registered, no longer receive Events from SipProvider.

- Client and Server Transaction creation methods.
  - For sending Request and Response messages statefully.

- CallIdHeader creation method.

- Send Requests and Responses statelessly.

- Listening Point manipulation methods.
  - Only one provider per listening point.

Responsibilities of JAIN SIP

- Provide methods to format SIP messages.

- The ability for an application to send and receive SIP messages.

- Parse incoming messages and enable application access to fields via a standardized Java interface.

- Invoke appropriate application handlers when protocol significant
  - Message arrivals and Transaction time-outs

- Provide Transaction support and manage Transaction state and lifetime on behalf of a user application.

- Provide Dialog support and manage Dialog state and lifetime on behalf on a user application.

SipListener Interface

- A single SipListener per SipStack which implies a single Listener in the architecture
  - All SipProviders associated to a Sipstack have the same SipListener.

- Process Request's either statefully or statelessly dependent on application logic.

- Process Response's to a recently sent Requests statefully.

- Process Transaction timeouts and retransmits Timer events.
  - Transaction processing notifications

Responsibilities of the Application

- Application registers an implementation of the SipListener interface to interact with the SIP Stack.

- Application must register with the SipProvider for all messaging capabilities with the stack.
  - Application requests transactions for stateful messaging.
  - Application sends stateless messages.
  - Access stack objects.

- Application receives messages from the stack as Events via the SipListener interface.

JAIN SIP Messaging Architecture
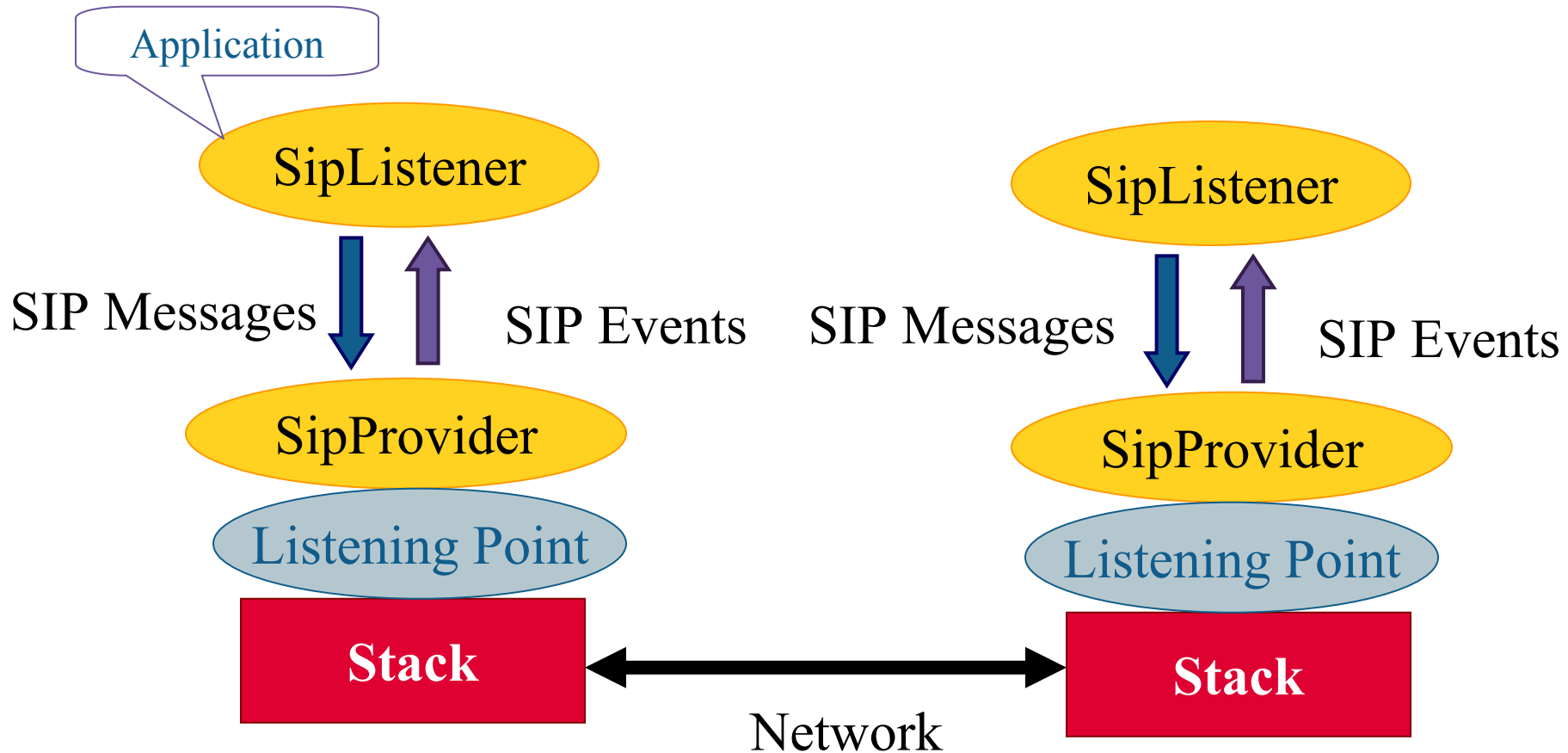
Event Model

- The architecture is developed for the J2SE environment therefore is event based utilizing the Listener/Provider event model.
  - There is a direct reference between the event provider and event consumer
  - Event consumer must register with the event provider
- Events encapsulate incoming Requests and Responses.
- Event Model is one way i.e. Application doesn't send out events, it sends out messages.
- The event model is asynchronous in nature using transactional identifiers to correlate messages.
- The SipListener represents the event consumer and listens for incoming Events that encapsulate messages that may be responses to initiated dialogs or new incoming dialogs.
- The SipProvider is the event provider who recieves messages from the network and passes them to the application as events.

Packages

- General package
  - Defines the architectural interfaces, the transaction and dialog interfaces and the event objects of the specification.

- Address package
  - Address package contains a generic URI wrapper and defines SIP URI and Tel URIs interfaces.

- Message package
  - Defines the interfaces necessary for the Request and Response messages.

- Header packages
  - Header package defines interfaces for all the supported headers and extension headers

Factories

JAIN SIP defines four different factories each with respective responsibilities, namely:

- SipFactory
  - This interface defines methods to create new Stack objects and other factory objects.

- AddressFactory
  - This interface defines methods to create SipURI's and TelURL's.

- HeaderFactory
  - This interface defines methods to create new Headers objects.

- MessageFactory
  - This interface defines methods to create new Request and Response objects.

Messages and Headers

Messages

- There are two type of messages in SIP, which JAIN SIP defines as Interfaces:
  - Request messages are sent from the client to server.
    - They contain a specific method type that identifies the type of Request.
    - A Request-URI which indicates the user or service to which this request is being addressed.
  - Response messages are sent from server to client in response to a Request.
    - They contain a specific status code that identifies the type of Response.
    - A Request-URI which indicates the user or service to which this request is being addressed.
    - A reason phrase that is intended for the human user.
- Messages may contain multiple Headers of the same type.
  - The Headers of a given type within a message is significant, i.e. Headers which are hop-by-hop must appear before any Headers which are end-to-end.
- A Message Body may contain a session description.
  - JAIN SIP defines this format an Object which allows the body to be a String or an Object type defined the Session Description Protocol (SDP) JSR specification and also a byte array.

Request Message Types

The following request messages are defined by the core SIP protocol:

- INVITE
  - Invites a participant to a session

- BYE
  - Ends a client's participation in a session

- CANCEL
  - terminates a transaction

- OPTIONS
  - Queries a participant about their media capabilities

- ACK
  - For reliability and call accepta nce (3-way handshake)

- REGISTER
  - Informs a SIP server about the location of a user

Request Message Types

The following request messages are defined by various SIP extensions:

- INFO

  - Session related control information generated during a session.

- PRACK

  - For reliability of provisional responses.

- UPDATE

  - Update a session without impacting the state of a dialog.

- SUBSCRIBE
  - Request notification from remote nodes when certain events occur.

- NOTIFY

  - Notification from remote nodes when certain events occur.

- MESSAGE
  - For sending instant messages.

- REFER
 - Refer to a resource provided in the request.

Headers

- SIP headers are similar to HTTP headers fields in both syntax and semantics.

- JAIN SIP models each SIP header as a specific interface as opposed to have a single generic interface to handle all header information.
  - Each interface specifies the Headers acceptable parameters.
  - More explicit protocol support – parsing support for each header.

- JAIN SIP supports all the headers defined in RFC 3261 and other headers introduced by supporting the following additional RFC's:
  - RFC3262 - RAckHeader and RSeqHeaders for the reliable delivery of provisional responses.
  - RFC3265 - AllowEventsHeader, EventHeader and SubscriptionStateHeader to support the event notification framework.
  - RFC3326 - ReasonHeader to support information on why the request was issued.
  - RFC3515 - ReferToHeader to support recipients to refer requests to another resource

JAIN SIP Extensible by Design

- SIP Extensions described in internet drafts and RFCs typically define:
    - New SIP Methods
        - New dialog creating methods
    - New SIP Headers.

- JAIN SIP defines an extensible framework to support new headers standardized for SIP:
    - New SIP methods can be set using the string method field of a request.
    - An application informs the stack of dialog creating methods, by specifying the method name to the EXTENSION_METHOD property of the SipStack configuration.

- JAIN SIP defines an extensible framework to support new headers standardized for SIP:
    - Defines a ExtensionHeader interface that contains the header name and header value attribute pair.
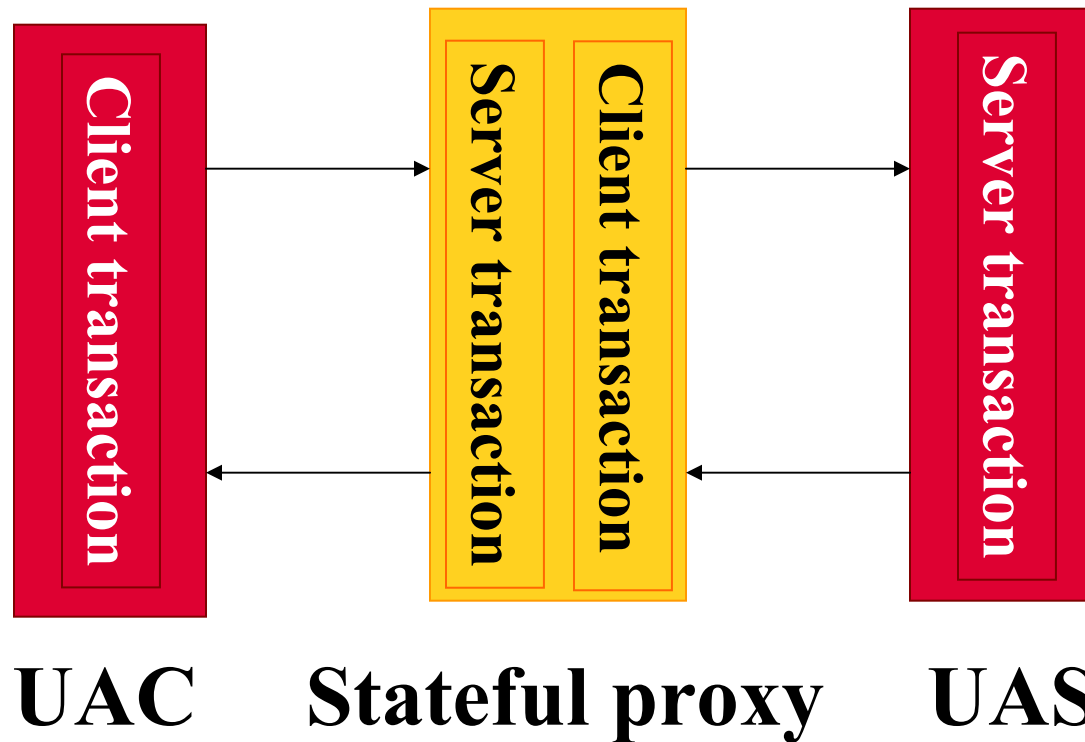   - Can be created and accessed by name.

Transactions and Dialogs

Generic SIP Application Structure
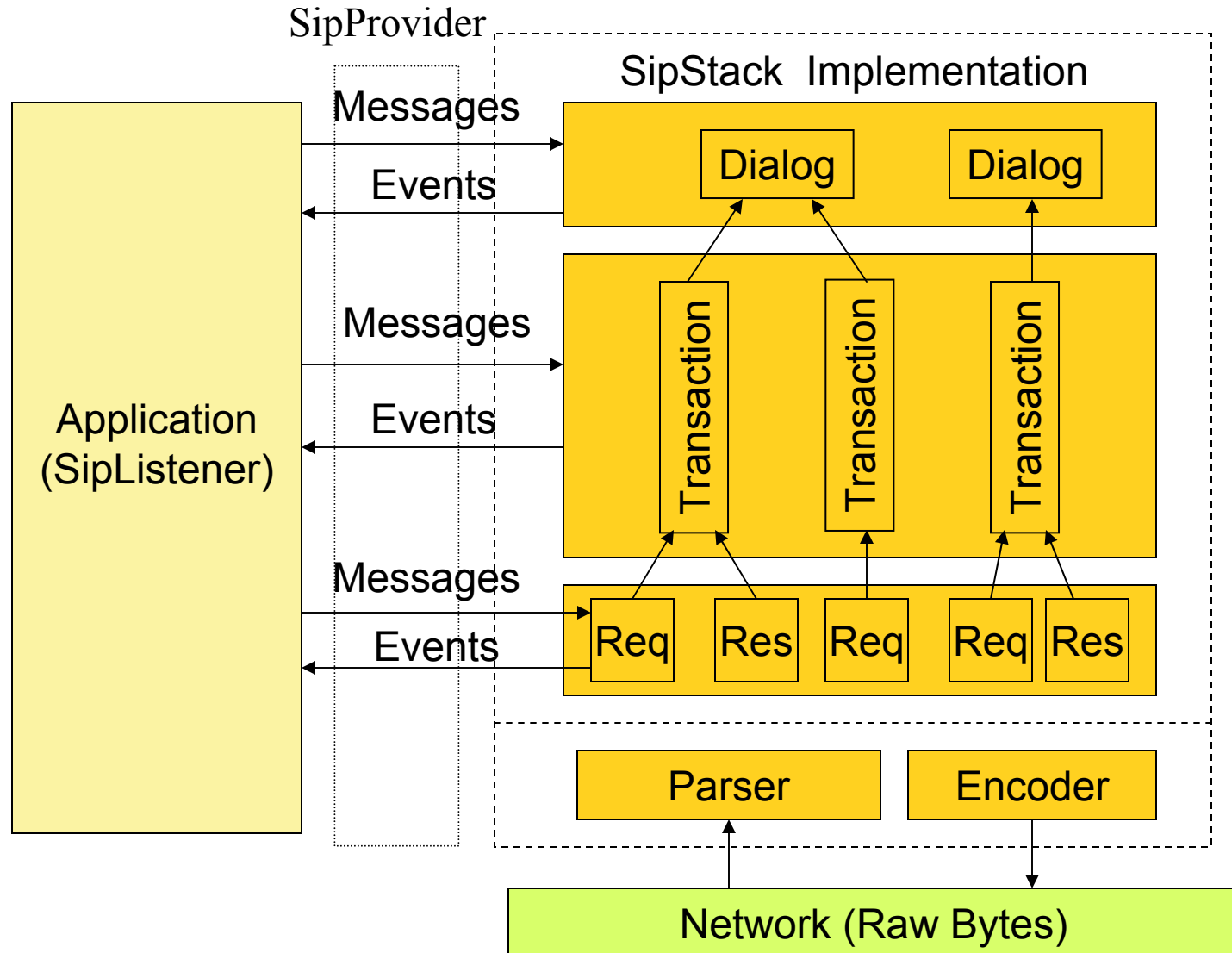
SIP Transactions

A SIP transaction consists of a single request and any responses to that request.

UAC **Stateful proxy** **UAS**

Transaction Support

- JAIN SIP standardizes the interface to the generic transactional model defined by the SIP protocol

  – JAIN SIP models both Client and Server Transactions as Interfaces.

- Transaction is created on incoming Request or may be created to send outgoing request.

  – When a Request is sent out statefully, application must request a ClientTransaction

  – When a new Request arrives, application determines whether to handle request via a ServerTransaction

  – When a Request in an existing dialog arrives the stack automatically associates it to a ServerTransaction

- When a response arrives, the Stack possibly associates a previously created ClientTransaction with the response

  – May be stray

- Messages are passed to the SipProvider in order to generate a new transaction. This transaction can be used to send the message onto the network

- Implementation manages the association between Transactions and Dialogs.
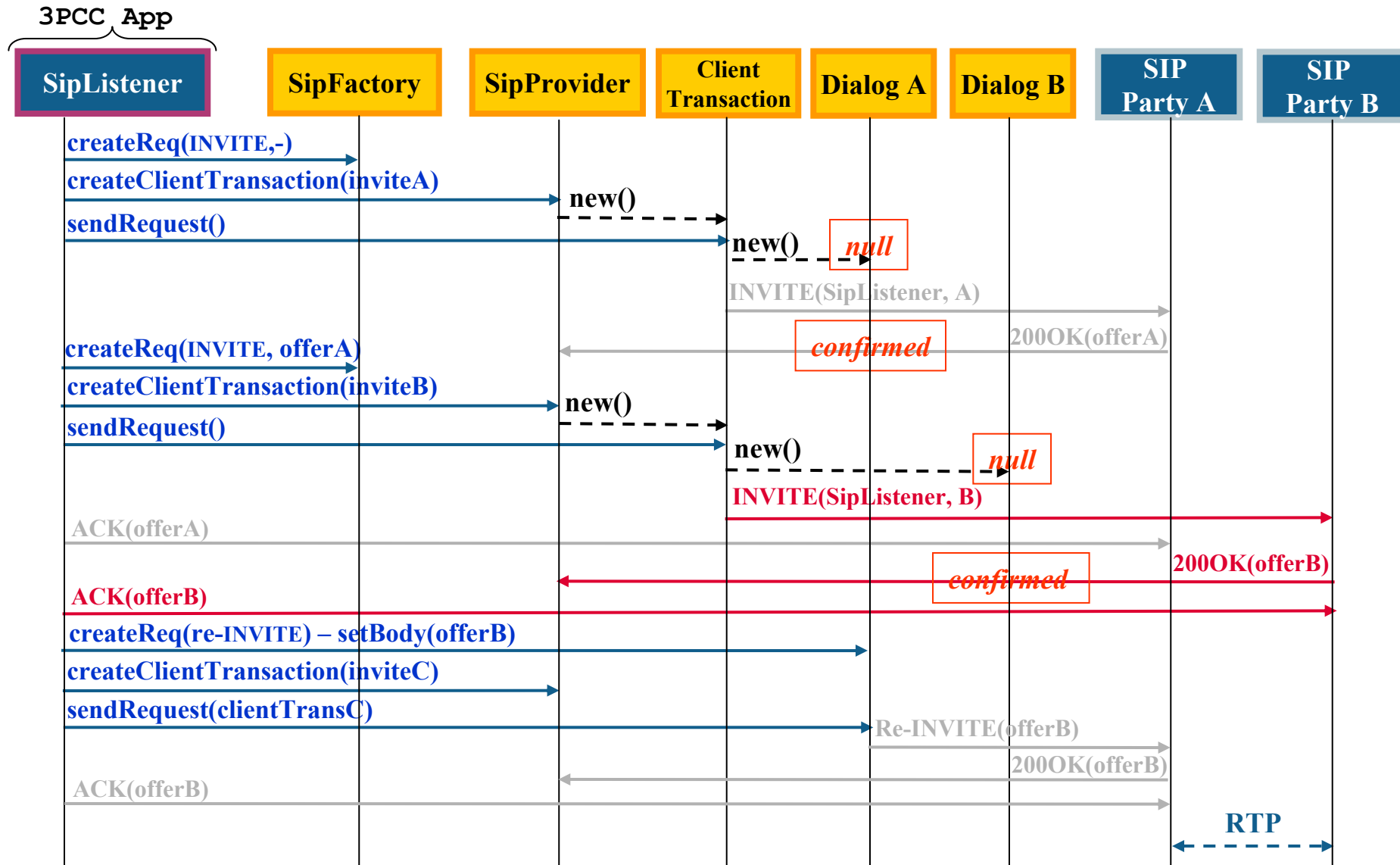
Dialog Support

- A Dialog is a peer to peer association between communicating SIP endpoints.
  - The dialog represents a context in which to interpret SIP messages.

- Dialogs are never directly created by the Application.
  - Dialogs are established by Dialog creating Transactions (INVITE, SUBSCRIBE…) and are managed by the stack.

- Dialog deletion may be under application control.
  - Though not generally recommended.

- Dialogs are used to maintain data needed for further message transmissions within the dialog
  - Route Sets, Sequence Numbers, URI's of the parties in the dialog.

- Dialogs have a state machine
  - Early, Confirmed, Completed and Terminated.

- Transactions may belong to a Dialog
  - Dialog state changes as a result of changes in Transaction State.
  - Access to dialog functionality from the transaction interface.

3PCC Example

Third Party Call Control – 3PCC

- 3PCC refers to the general ability to establish and manipulate calls between other parties.

- Establishment of these calls is orchestrated by a third party, referred to as the controller:
  - A controller is a SIP User Agent that wishes to create a session between two other user agents.

- 3PCC is often used for:
  - operator services i.e. the operator creates a call that connects two participants together.
  - conferencing.

3PCC Example using JAIN SIP

3PCC App

SipListener	SipFactory	SipProvider	Client Transaction	Dialog A	Dialog B	SIP Party A	SIP Party B

createReq(INVITE,-)

createClientTransaction(inviteA)

sendRequest() → new()

new() - - → *null*

INVITE(SipListener, A)

200OK(offerA)

createReq(INVITE, offerA) ← *confirmed*

createClientTransaction(inviteB) → new()

sendRequest()

new()

- - → *null*

INVITE(SipListener, B)

ACK(offerA)

200OK(offerB)

confirmed

ACK(offerB)

createReq(re-INVITE) – setBody(offerB)

createClientTransaction(inviteC)

sendRequest(clientTransC)

Re-INVITE(offerB)

200OK(offerB)

ACK(offerB)

RTP

Latest Specification Updates

JAIN SIP v1.0

- RFC2543 Supported.

- J2SE 1.3 and above.

- Transactions referenced by long.

- Transaction state is not visible to application.

- No explicit Dialog Support.

- Stack Configuration not defined.

JAIN SIP v1.1

- RFC3261 Supported.

- J2SE 1.4 and above.

- Transaction interfaces defined.

- Transaction/Dialog state can be read by application.

- Dialog interface defined and managed by stack.

- Stack Configured with defined properties.

Presentation Outline

- What is SIP?

- Why create JAIN SIP?

- Introduction to JAIN SIP

- Developer Code Snippets

- Implementation Used-Cases

Application - Stack Creation

Initialize Stack using SipFactory:

```java
try {
    Properties properties = new Properties();
    properties.setProperty("javax.sip.IP_ADDRESS",
                           "129.6.55.181");

    properties.setProperty("javax.sip.OUTBOUND_PROXY",
                       "129.6.55.182:5070/UDP");

    ……// Other initialization properties.
    try {
        sipStack = sipFactory.createSipStack(properties);
    } catch(SipException e) {
            System.exit(-1);
    }
}
```

Application – Request Creation

Initialize Request using Factories:

```
try {
   SipURI requestURI = addressFactory.createSipURI
                          (toUser, toSipAddress);
   // … Create other headers
   Request request = messageFactory.createRequest
        (requestURI, Request.INVITE, callIdHeader,
        cSeqHeader, fromHeader, toHeader,
        viaHeaders, maxForwards);
}
```

Application - Sending Requests

Send outgoing messages:

```
try {
    // Create the client transaction
    ClientTransaction inviteTid =
        sipProvider.getNewClientTransaction(request);
    // send the request
    inviteTid.sendRequest();
}
```

Application – Processing Requests

Handle incoming messages as Events:

```
try {
  public void  processRequest(RequestEvent
      requestEvent) {
      Request request =
              requestEvent.getRequest();
      ServerTransaction st =
              requestEvent.getServerTransaction();
      // do request specific processing here
  }
}
```

Presentation Outline

- What is SIP?

- Why create JAIN SIP?

- Introduction to JAIN SIP

- Developer Code Snippets

- Implementation Used-Cases

JAIN SIP for Instant Messaging

- Suitable for building IM and Presence Clients and Servers.

- API supports the required methods and Headers.

- Creates and manages Dialogs for SUBSCRIBE and MESSAGE methods.

- NIST-SIP JAIN IM Client SipListener is about 1100 LOC.

- Interoperates with Microsoft Messenger IM

http://jain-sip-presence-proxy.dev.java.net

JAIN SIP for Proxy Servers

- Facilities construction of Proxy Servers
  - Stateless, Transaction-stateful, and Dialog-stateful operation.

- Access to Dialog/Transaction state and route tables.

- Extensibility and application controlled Routing.

- Deep copy semantics for cloning.

- Incorporates IM + Presence Support
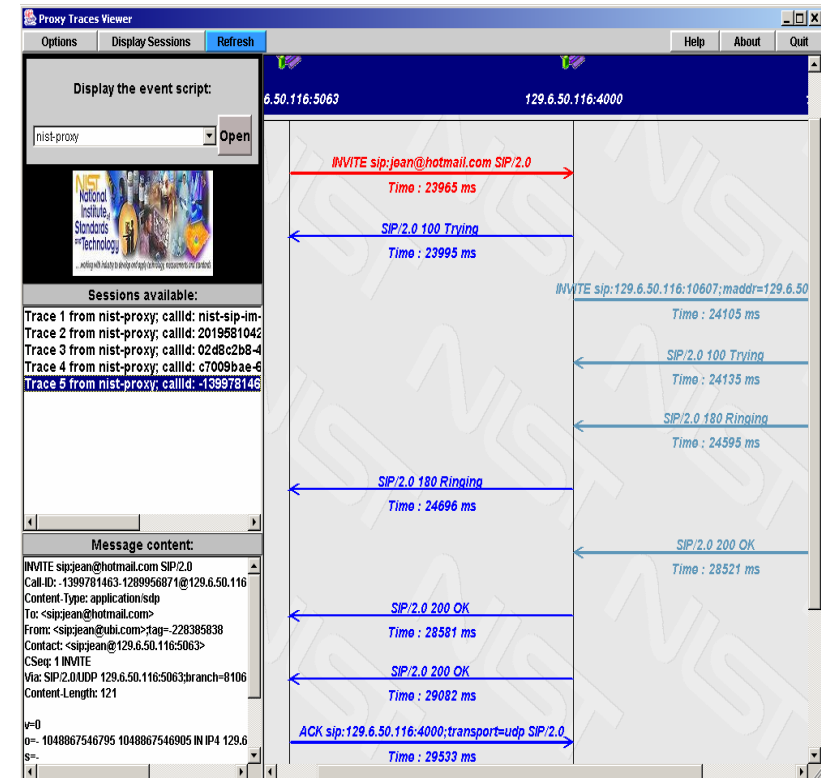
http://jain-sip-presence-proxy.dev.java.net

JAIN SIP for Telephony

- Ideal for building telephony applications.

- API provides a complete set of functionality for managing calls.

- Spares the application the burden of managing dialogs and transactions.

- SIP COMMUNICATOR - a complete example of an audio/video telephony application (uses JAIN SIP RI and JMF)

  http://sip-communicator.dev.java.net

- Interoperates with Microsoft Windows Messenger.
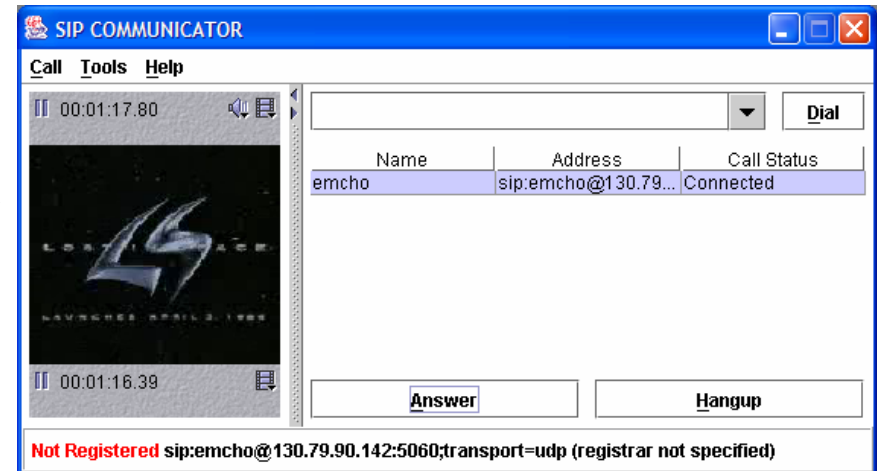
JAIN SIP Reference Implementation

- In the public domain.
  - Includes example IM client and proxy.
  - Includes trace visualization tools.

- Footprint
  - About 46000 LOC.
  - Jar file about 355 Kb
  - 3Mb of memory after running a few requests.

http://www-x.antd.nist.gov/proj/iptel

JAIN SIP Resources

- JAIN SIP Specification:
  http://jcp.org/jsr/detail/032.jsp

- JAIN SIP Discussion List:
  http://archives.java.sun.com/jain-sip-interest.html

- JAIN SIP Collaboration Project:
  http://jain-sip.dev.java.net

- SIP-Communicator Collaboration Project:
  http://sip-communicator.dev.java.net

- SIP-Presence-Proxy Collaboration Project:
  http://jain-sip-presence-proxy.dev.java.net

JSR 32
http://jcp.org/en/jsr/detail?id=32

Subscribe to:
http://archives.java.sun.com/jain-sip-interest.html